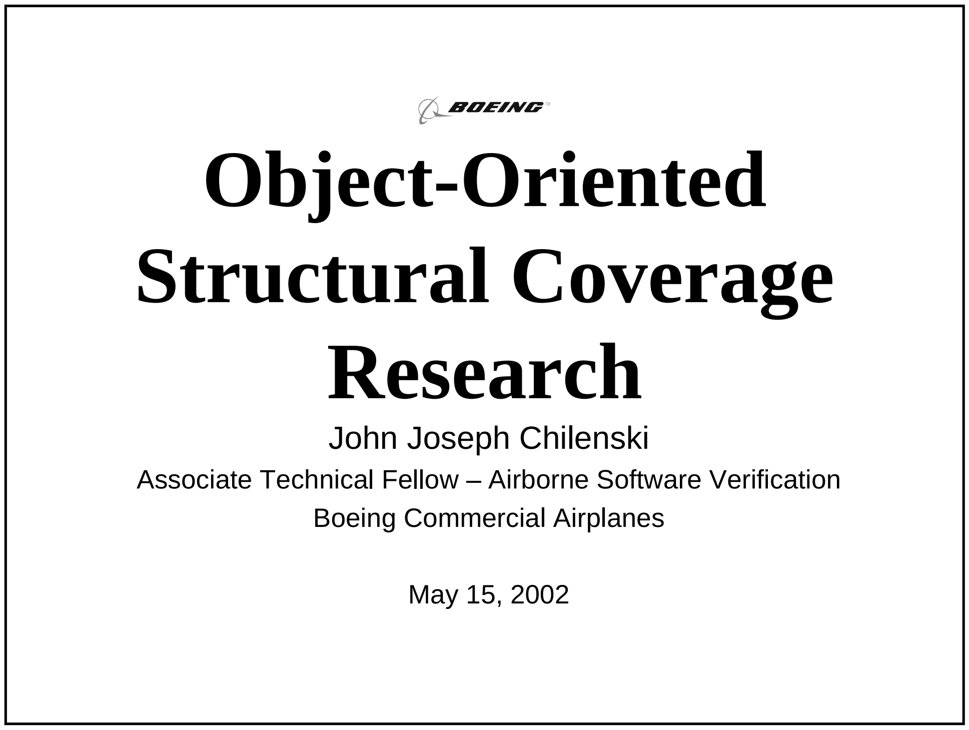
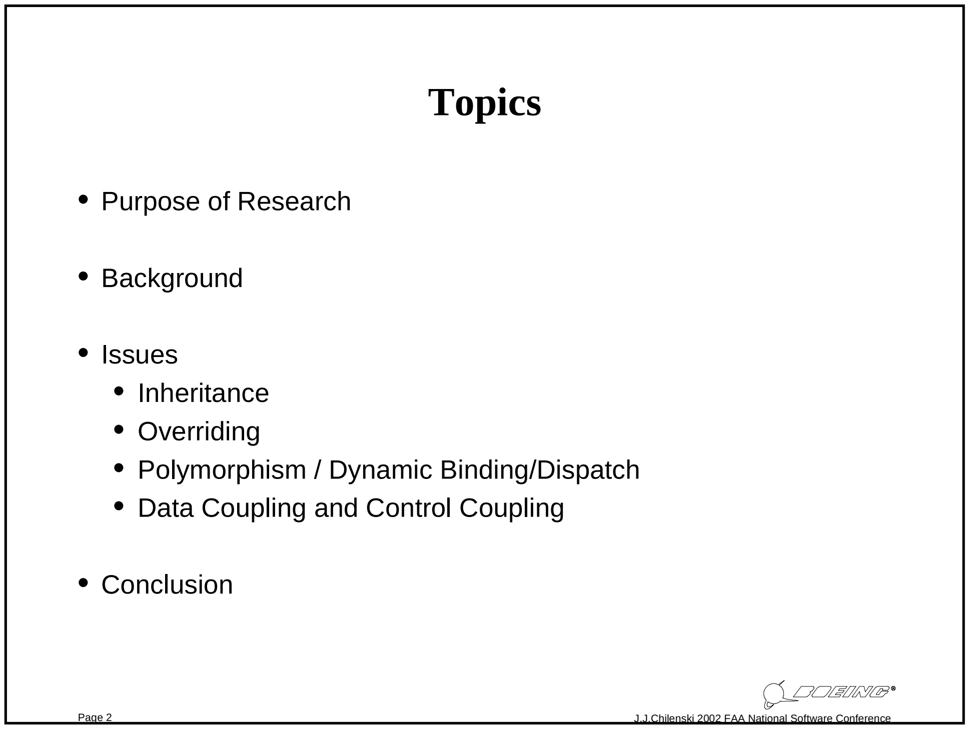


Object-Oriented Structural Coverage Research

John Joseph Chilenski
Associate Technical Fellow – Airborne Software Verification
Boeing Commercial Airplanes

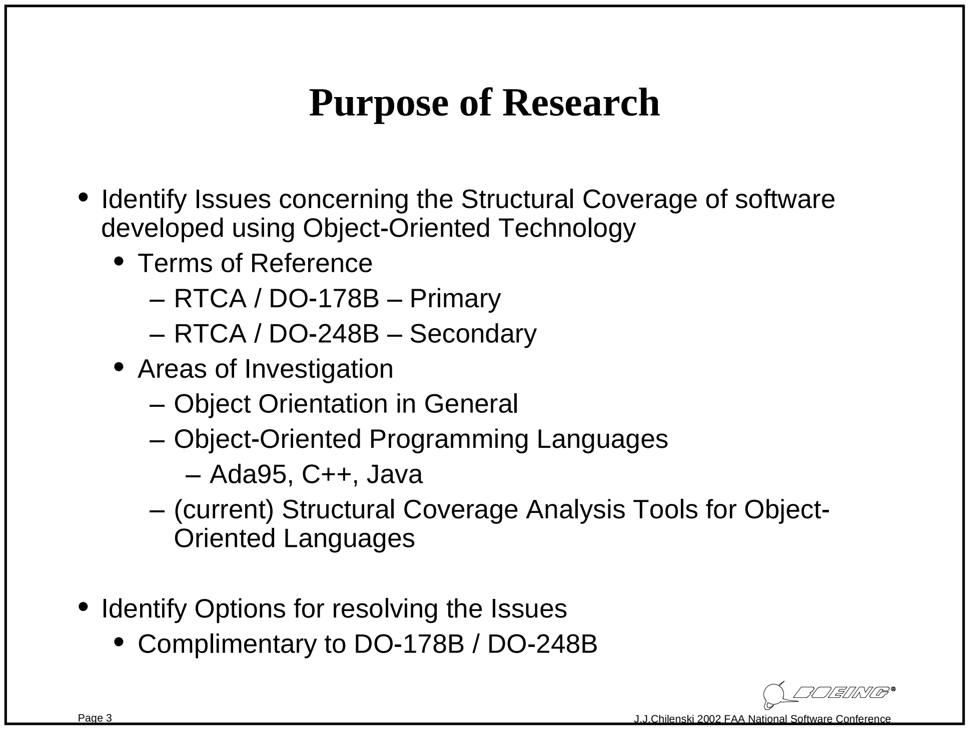
May 15, 2002



Topics

- Purpose of Research
- Background
- Issues
  - Inheritance
  - Overriding
  - Polymorphism / Dynamic Binding/Dispatch
  - Data Coupling and Control Coupling
- Conclusion

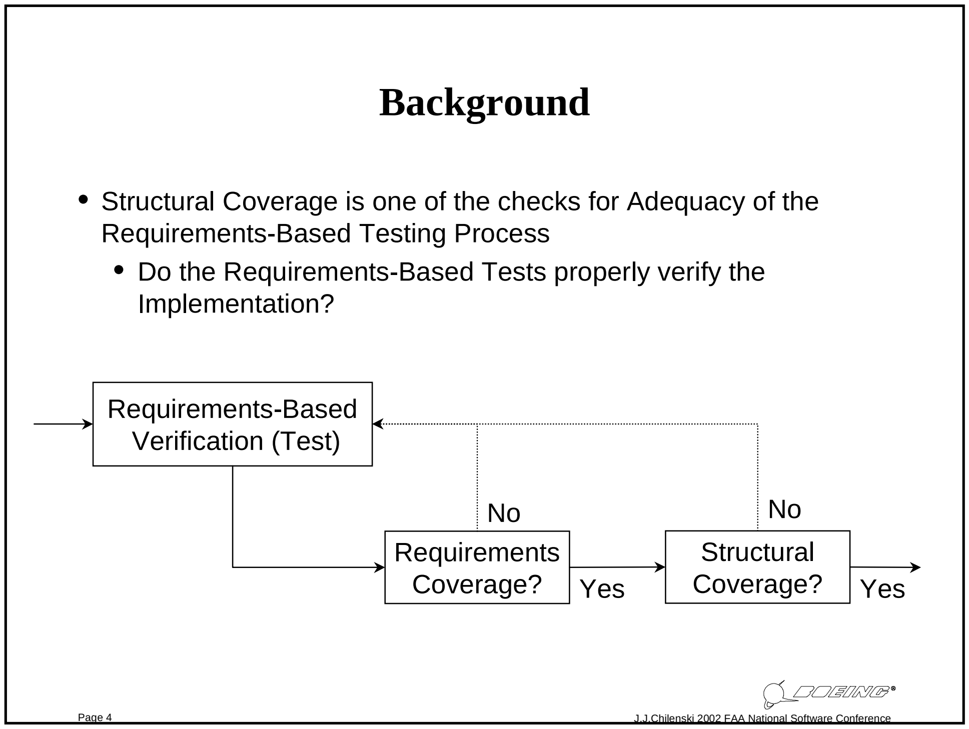
Page 2 J.J.Chilenski 2002 FAA National Software Conference



Purpose of Research

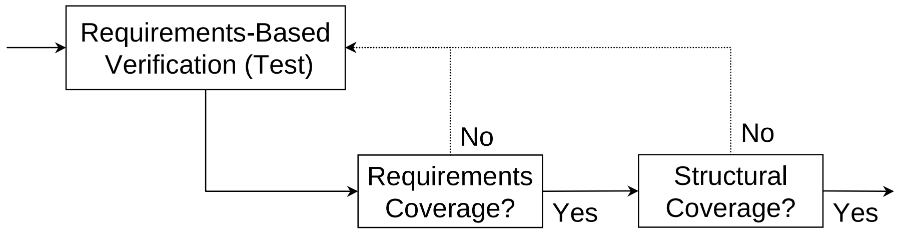
- Identify Issues concerning the Structural Coverage of software developed using Object-Oriented Technology
  - Terms of Reference
    - RTCA / DO-178B – Primary
    - RTCA / DO-248B – Secondary
  - Areas of Investigation
    - Object Orientation in General
    - Object-Oriented Programming Languages
      - Ada95, C++, Java
    - (current) Structural Coverage Analysis Tools for Object-Oriented Languages
- Identify Options for resolving the Issues
  - Complimentary to DO-178B / DO-248B

Page 3 J.J.Chilenski 2002 FAA National Software Conference



Background

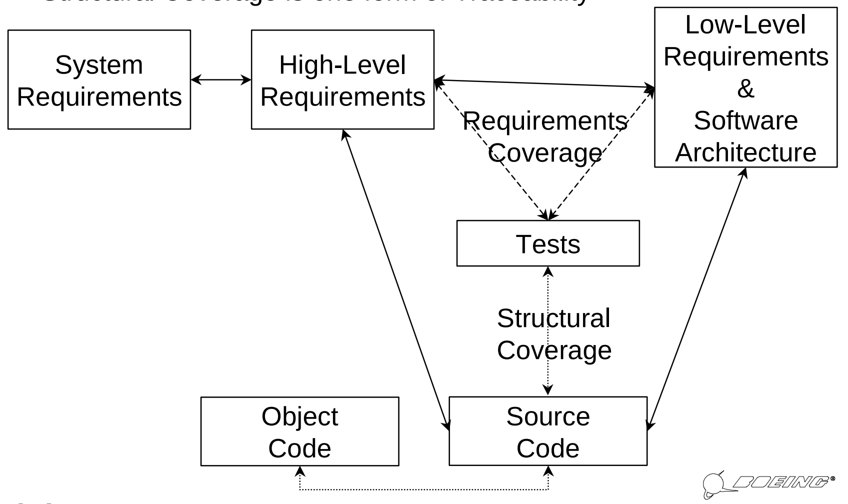
- Structural Coverage is one of the checks for Adequacy of the Requirements-Based Testing Process
  - Do the Requirements-Based Tests properly verify the Implementation?



Page 4 J.J.Chilenski 2002 FAA National Software Conference

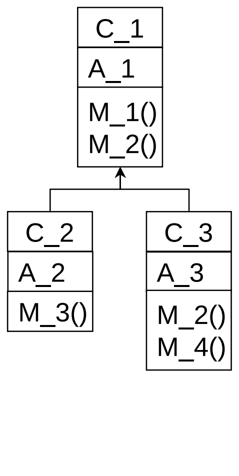
Background (continued)

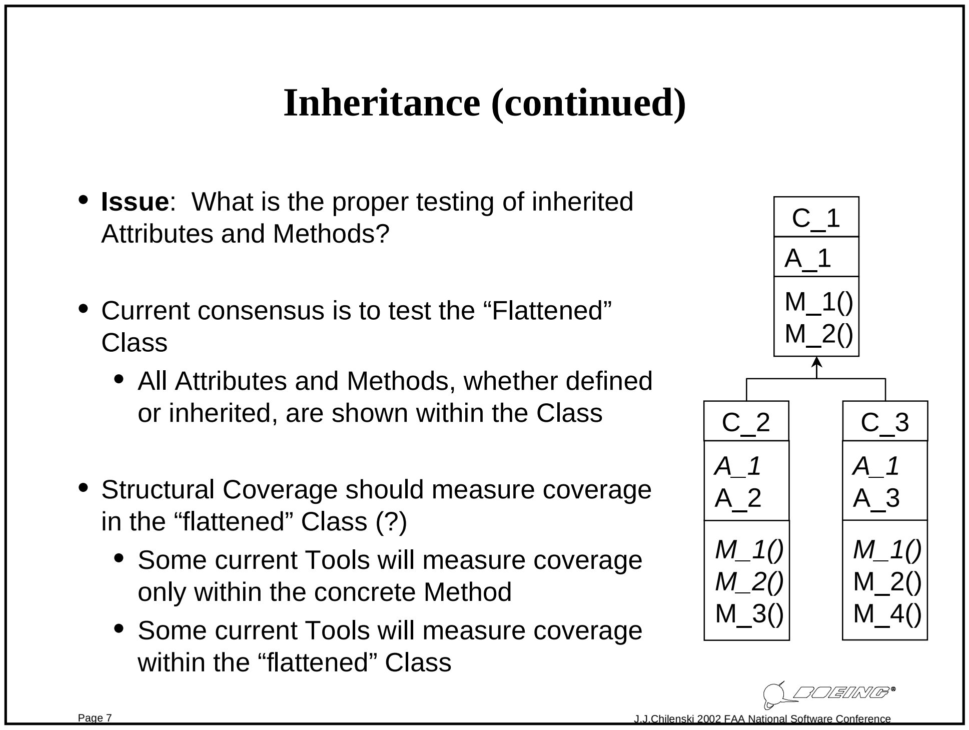
- Structural Coverage is one form of Traceability



Inheritance

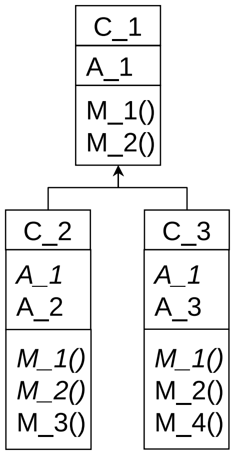
- One of the fundamental building blocks of OOT
- Mechanism whereby a Class is defined in terms of other Classes
  - **Extension** is the inclusion of the operations, attributes and methods of ancestor classes (parents, their parents, etc.) in a subclass
  - **Overriding** is the definition of either an attribute or method in a class with the same signature as that in an ancestor class
  - **Specialization** is the definition of operations, attributes and methods that are unique to that class



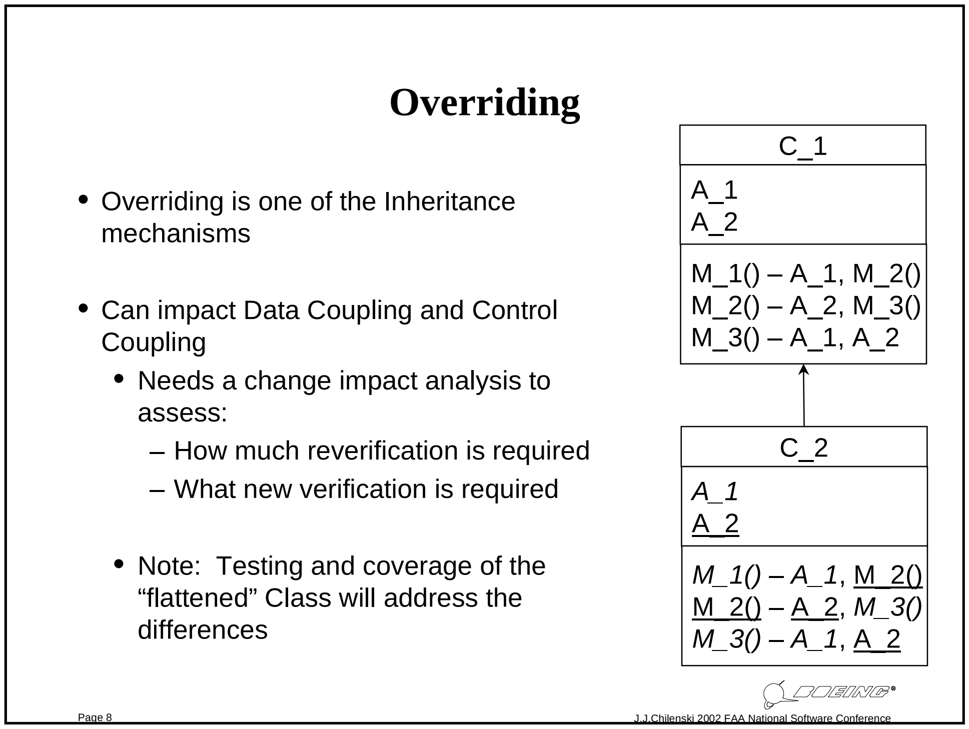


Inheritance (continued)

- **Issue**: What is the proper testing of inherited Attributes and Methods?
- Current consensus is to test the “Flattened” Class
  - All Attributes and Methods, whether defined or inherited, are shown within the Class
- Structural Coverage should measure coverage in the “flattened” Class (?)
  - Some current Tools will measure coverage only within the concrete Method
  - Some current Tools will measure coverage within the “flattened” Class

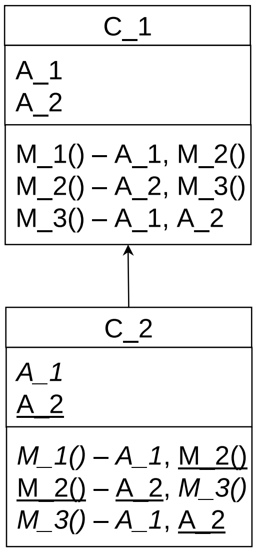


Page 7 J.J.Chilenski 2002 FAA National Software Conference

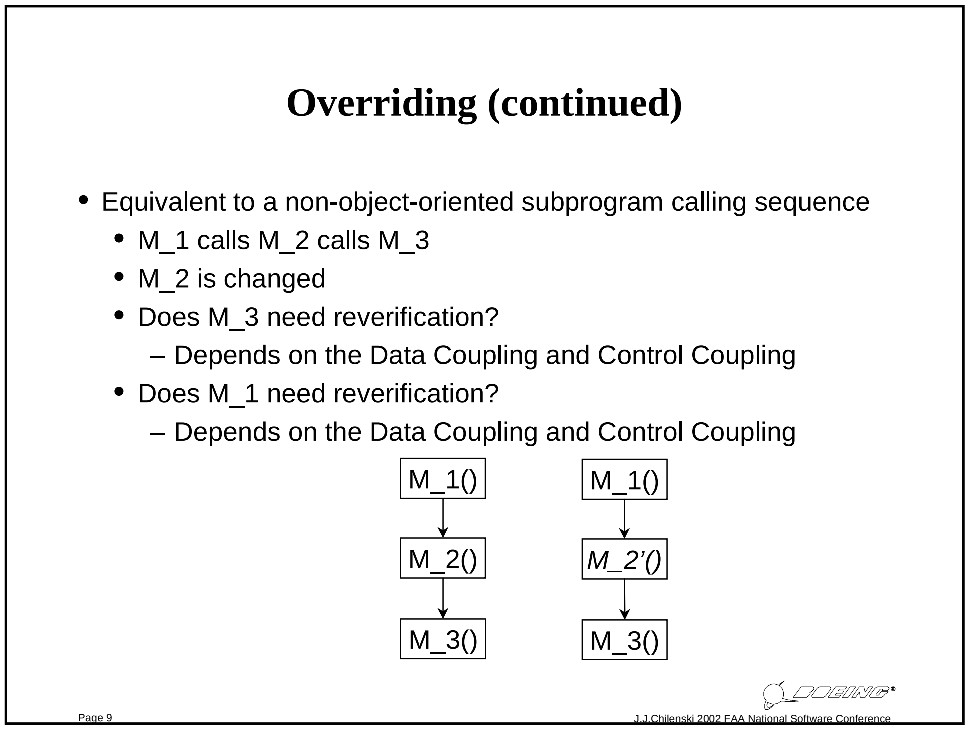


Overriding

- Overriding is one of the Inheritance mechanisms
- Can impact Data Coupling and Control Coupling
  - Needs a change impact analysis to assess:
    - – How much reverification is required
    - – What new verification is required
  - Note: Testing and coverage of the “flattened” Class will address the differences



Page 8 J.J.Chilenski 2002 FAA National Software Conference

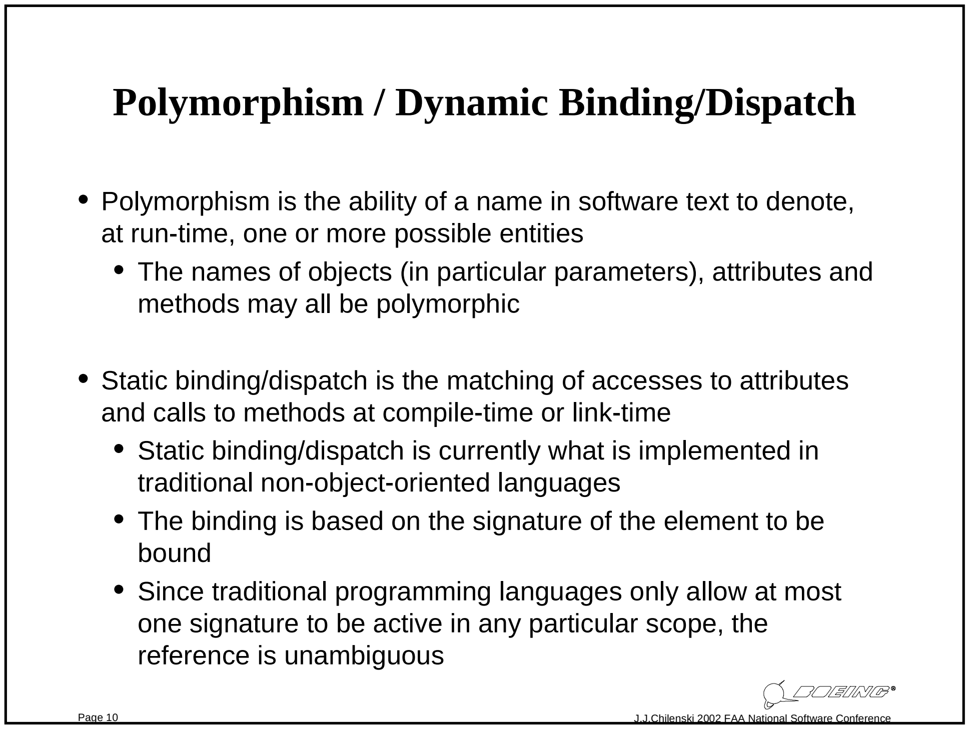


Overriding (continued)

- Equivalent to a non-object-oriented subprogram calling sequence
  - M_1 calls M_2 calls M_3
  - M_2 is changed
  - Does M_3 need reverification?
    - – Depends on the Data Coupling and Control Coupling
  - Does M_1 need reverification?
    - – Depends on the Data Coupling and Control Coupling

$$\begin{tikzcd}
\boxed{\text{M\_1()}} \arrow[d] & \boxed{\text{M\_1()}} \arrow[d] \\
\boxed{\text{M\_2()}} \arrow[d] & \boxed{\textit{M\_2'()}} \arrow[d] \\
\boxed{\text{M\_3()}} & \boxed{\text{M\_3()}}
\end{tikzcd}$$

Page 9 J.J.Chilenski 2002 FAA National Software Conference



Polymorphism / Dynamic Binding/Dispatch

- Polymorphism is the ability of a name in software text to denote, at run-time, one or more possible entities
  - The names of objects (in particular parameters), attributes and methods may all be polymorphic

- Static binding/dispatch is the matching of accesses to attributes and calls to methods at compile-time or link-time
  - Static binding/dispatch is currently what is implemented in traditional non-object-oriented languages
  - The binding is based on the signature of the element to be bound
  - Since traditional programming languages only allow at most one signature to be active in any particular scope, the reference is unambiguous

Page 10 J.J.Chilenski 2002 FAA National Software Conference

Polymorphism / Dynamic Binding/Dispatch (continued)

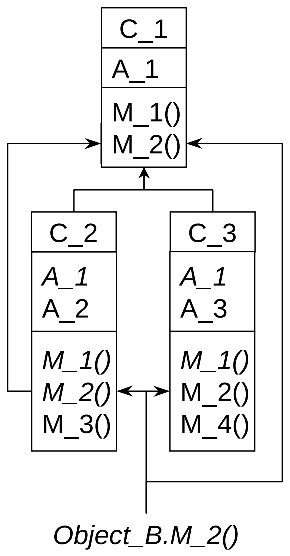
- Dynamic binding/dispatch is the matching of accesses to attributes and calls to methods at run-time as opposed to compile-time or link-time
  - This results from a polymorphic reference or call
    - – Example already seen in Overriding
  - In essence what is happening is that one of the key components of the signature, namely the class the object belongs to at the time of execution, is missing from the signature
    - – Needs to be determined at run-time

- Note that it is possible to have both static and dynamic binding/dispatch present in OOT software/systems

Polymorphism / Dynamic Binding/Dispatch (continued)

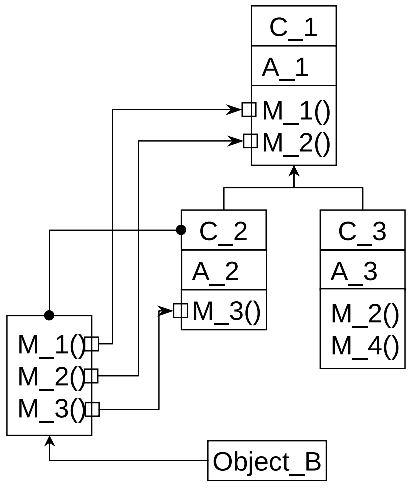
- For static binding/dispatch, it is sufficient to record coverage of the access and/or call statement itself, as that access and/or call never changes
  - The call to *Object_A.M_3()* is unambiguous
  - There will appear a call to *C_2.M_3* in the object code

- For dynamic binding/dispatch, something more may be needed
  - The call to *Object_B.M_2() is ambiguous*
  - It is not clear whether *C_1.M_2()* or *C_3.M_2()* is to be called
    - – The appropriate method to call depends on the type (class) of *Object_B*



Polymorphism / Dynamic Binding/Dispatch (continued)

- In essence, the compiler/linker generates a “case statement” to determine the class of Object_B and call the appropriate method
  - Most implementations use Method (dispatch) Tables
  - Every Object points to its class methods table
  - Compiler generates lookup code for the jump table
    – Inlined dispatch routine



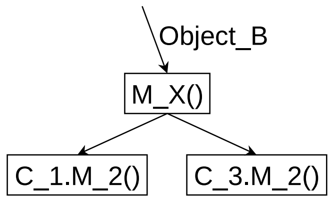
Polymorphism / Dynamic Binding/Dispatch (continued)

- **Issue**: What is the proper testing of polymorphic references (dynamic binding/dispatch)?
  - No current consensus

- **Issue**: How should structural coverage of polymorphic references (dynamic binding/dispatch) be recorded?
  - No current consensus
  - Some current Tools consider execution of the polymorphic reference sufficient (Statement Coverage of Source)
  - Some current Tools consider execution of every possible resolution at every polymorphic reference sufficient (Decision Coverage of Object)
  - Some current Tools consider execution of every polymorphic reference and every entry in every Method Table sufficient

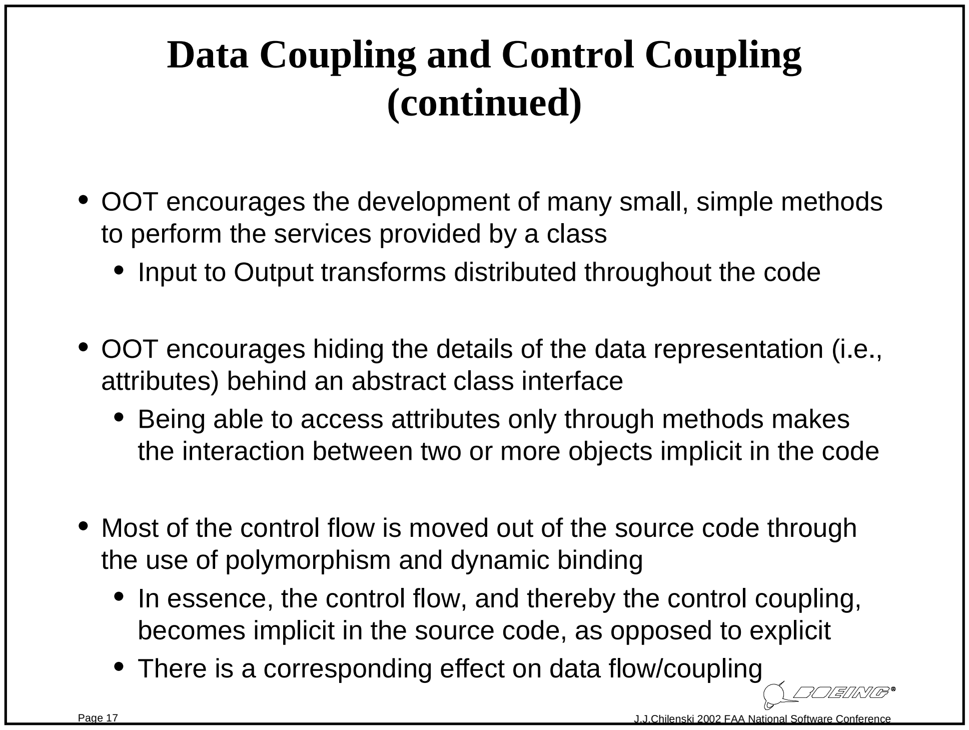
Polymorphism / Dynamic Binding/Dispatch (continued)

- At the very least, confirmation of Data Coupling and Control Coupling is impacted by Dynamic Binding/Dispatch
  - Equivalent to a non-object-oriented subprogram making conditional calls based on the value of a parameter



Data Coupling and Control Coupling

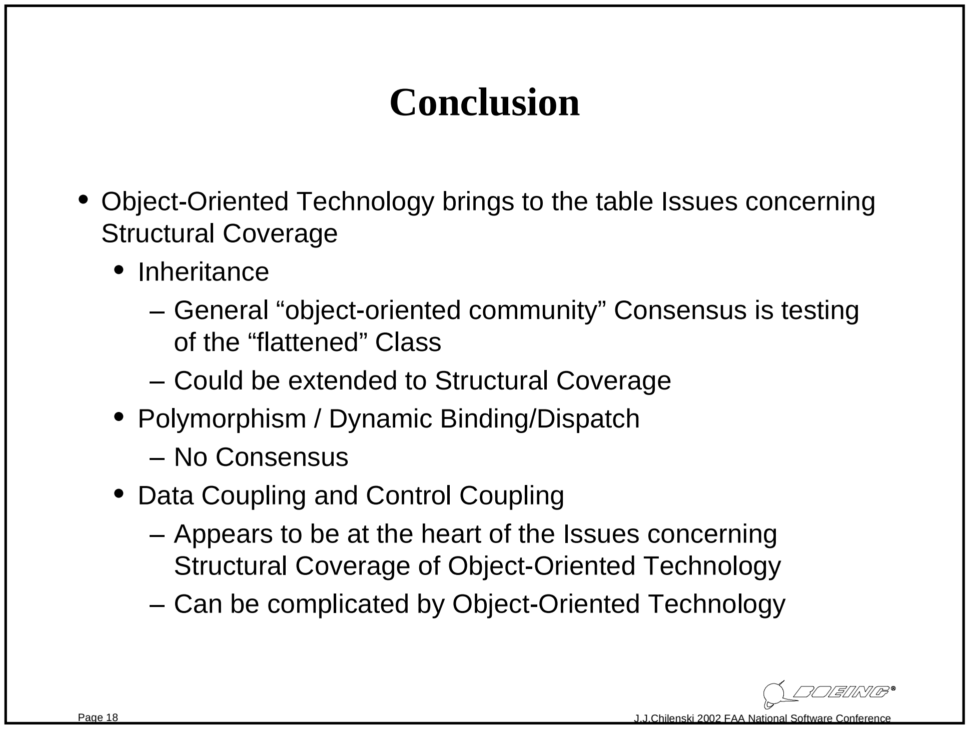
- Data Coupling and Control Coupling are not new to Object-Oriented Technology
- Already identified some specific Issues regarding Data Coupling and Control Coupling due to Inheritance, Polymorphism and Dynamic Binding/Dispatch
- Object-Oriented Technology introduces some broader Issues



Data Coupling and Control Coupling (continued)

- OOT encourages the development of many small, simple methods to perform the services provided by a class
  - Input to Output transforms distributed throughout the code
- OOT encourages hiding the details of the data representation (i.e., attributes) behind an abstract class interface
  - Being able to access attributes only through methods makes the interaction between two or more objects implicit in the code
- Most of the control flow is moved out of the source code through the use of polymorphism and dynamic binding
  - In essence, the control flow, and thereby the control coupling, becomes implicit in the source code, as opposed to explicit
  - There is a corresponding effect on data flow/coupling

Page 17 J.J.Chilenski 2002 FAA National Software Conference



Conclusion

- Object-Oriented Technology brings to the table Issues concerning Structural Coverage
  - Inheritance
    - General “object-oriented community” Consensus is testing of the “flattened” Class
    - Could be extended to Structural Coverage
  - Polymorphism / Dynamic Binding/Dispatch
    - No Consensus
  - Data Coupling and Control Coupling
    - Appears to be at the heart of the Issues concerning Structural Coverage of Object-Oriented Technology
    - Can be complicated by Object-Oriented Technology

Page 18 J.J.Chilenski 2002 FAA National Software Conference